# MỤC LỤC

Trang

Trang phụ bìa ..i

Lời cam đoan .ii

Lời cảm ơn iii

[**MỤC LỤC** 1](#_Toc401913506)

[DANH MỤC CÁC CHỮ VIẾT TẮT 4](#_Toc401913507)

[DANH MỤC CÁC BẢNG 4](#_Toc401913508)

[DANH MỤC CÁC SƠ ĐỒ 4](#_Toc401913509)

[MỞ ĐẦU 5](#_Toc401913510)

[Lý do chọn đề tài (tính cấp thiết của đề tài) 5](#_Toc401913511)

[Mục đích nghiên cứu của đề tài 5](#_Toc401913512)

[Đối tượng và phạm vi nghiên cứu 5](#_Toc401913513)

[Phương pháp nghiên cứu 5](#_Toc401913514)

[Kết cấu của đề tài 5](#_Toc401913515)

[CHƯƠNG 1: TIẾP NHẬN VÀ XỬ LÝ YÊU CẦU 6](#_Toc401913516)

[1.1. Bản mô tả yêu cầu 6](#_Toc401913517)

[1.2. Yêu cầu chức năng nghiệp vụ 6](#_Toc401913518)

[1.3. Yêu cầu kỹ thuật 6](#_Toc401913519)

[1.3.1. Yêu cầu chung về hệ thống thông tin 6](#_Toc401913520)

[1.3.2. Yêu cầu cụ thể 7](#_Toc401913521)

[1.3.2.1. Yêu cầu phần cứng 7](#_Toc401913522)

[1.3.2.2. Yêu cầu phần mềm 7](#_Toc401913523)

[CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 8](#_Toc401913524)

[2.1. Hướng phân tích và thiết kế 8](#_Toc401913525)

[2.1. Lược đồ quan niệm 8](#_Toc401913526)

[2.2. Lược đồ trường hợp sử dụng (Use cases) 8](#_Toc401913527)

[2.3. Lược đồ lớp (class) 8](#_Toc401913528)

[2.3.1. Lược đồ các lớp thực thể 8](#_Toc401913529)

[2.3.1.1. Các giao diện (interface) và lớp ảo (abstract) 9](#_Toc401913530)

[2.3.1.2. Các lớp thực thể 12](#_Toc401913531)

[2.3.1. Lược đồ các lớp thư viện liên quan 27](#_Toc401913532)

[2.4. Lược đồ CSDL mức vật lý 27](#_Toc401913533)

[2.5. Lược đồ tuần tự (sequences) 28](#_Toc401913534)

[2.5.1. 28](#_Toc401913535)

[2.5.2. 28](#_Toc401913536)

[2.5.3. 28](#_Toc401913537)

[2.5.4. 28](#_Toc401913538)

[2.5.5. 28](#_Toc401913539)

[CHƯƠNG 3: THỰC THI 29](#_Toc401913540)

[3.1. Môi trường lập trình và phát triển ứng dụng 29](#_Toc401913541)

[3.2. Mô hình triển khai và cách tổ chức ứng dụng 29](#_Toc401913542)

[3.2.1. Mô hình đa nền tảng (cross-platform) trong ứng dụng hệ .NET 29](#_Toc401913543)

[3.2.2. Mô hình 3 lớp trong lập trình hướng đối tượng (OOP) 29](#_Toc401913544)

[3.2.3. Mô hình MVP (Model-View-Presenter) Winform dành cho ứng dụng Desktop 29](#_Toc401913545)

[3.2.4. Mô hình ASP.NET Webform dành cho ứng dụng Web 30](#_Toc401913546)

[3.3. Các công nghệ và kỹ thuật lập trình được áp dụng 30](#_Toc401913547)

[3.3.1. Công nghệ Entity Framework (EF) trong lập trình dữ liệu hướng đối tượng (OOP) 30](#_Toc401913548)

[3.3.1.1. Tổng quan 30](#_Toc401913549)

[3.3.1.2. Mô hình triển khai Code first 30](#_Toc401913550)

[3.3.1.3. Ánh xạ TPC (Table Per Concrete class) trong kế thừa thuộc tính 31](#_Toc401913551)

[3.3.1.4. Tích Đề-các (Descartes) trong biểu diễn quan hệ 1-n hoặc n-n 31](#_Toc401913552)

[3.3.1.5. Công nghệ truy vấn LINQ 32](#_Toc401913553)

[3.3.1.6. Các tính năng khác được áp dụng trong đề tài 33](#_Toc401913554)

[3.3.2. Công nghệ Sync Framework của Microsoft trong đồng bộ CSDL tập trung 39](#_Toc401913555)

[3.3.2.1. Tổng quan 39](#_Toc401913556)

[3.3.2.2. Đồ thị tiến trình đồng bộ trong kỹ thuật tránh deadlock 40](#_Toc401913557)

[3.3.2.3. Tiến trình song song trong đồ thị dạng cây 44](#_Toc401913558)

[3.3.2.4. Đụng độ dữ liệu và giải pháp khóa chính GUID 44](#_Toc401913559)

[3.3.2.5. Kỹ thuật khóa Semaphore trong xử lý đa luồng (multi thread) trên hàng đợi (queue) 46](#_Toc401913560)

[3.3.4. Công nghệ DevExpress trong lập trình giao diện 48](#_Toc401913561)

[3.3.5. Công nghệ giao diện tùy biến (responsive design) dành cho ứng dụng Web Mobile 48](#_Toc401913562)

[CHƯƠNG 4: KIỂM THỬ 49](#_Toc401913563)

[4.1. Kiểm thử tự động mức mã nguồn (Unit test) 49](#_Toc401913564)

[4.1.1. Kiểm thử hộp đen (Black box testing) 49](#_Toc401913565)

[4.1.2. Mô hình kiểm thử AAA (Arrange-Act-Assert) 49](#_Toc401913566)

[4.1.3. Kiểm thử các thành phần phụ thuộc (Dependent module) 49](#_Toc401913567)

[4.2. Kiểm thử chấp nhận (Acceptance test) 49](#_Toc401913568)

[CHƯƠNG 5: VẬN HÀNH VÀ BẢO TRÌ 51](#_Toc401913569)

[5.1. Hướng dẫn sử dụng và tài liệu kỹ thuật 51](#_Toc401913570)

[5.2. Vận hành hệ thống 51](#_Toc401913571)

[5.3. Bảo trì 51](#_Toc401913572)

[KẾT LUẬN VÀ KIẾN NGHỊ 52](#_Toc401913573)

[TÀI LIỆU THAM KHẢO 53](#_Toc401913574)

DANH MỤC CÁC CHỮ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| CSDL | : | Cơ sở dữ liệu |
| EF | : | Entity Framework |
|  |  |  |
|  |  |  |

DANH MỤC CÁC BẢNG

|  |  |  |
| --- | --- | --- |
| **TT** | **Tên bảng** | **Trang** |
| 1 | Bảng 3.1. Tên bảng | 48 |
| 2 | Bảng 3. | 48 |

DANH MỤC CÁC SƠ ĐỒ

|  |  |  |
| --- | --- | --- |
| **TT** | **Tên sơ đồ** | **Trang** |
| 1 | Bảng 3.1. Tên bảng | 48 |
|  | Hình 3.x: Minh họa semaphore đơn tiến trình trong xử lý đa luồng trên hàng đợi |  |
| 2 | Bảng 3. | 48 |

MỞ ĐẦU

## Lý do chọn đề tài (tính cấp thiết của đề tài)

## Mục đích nghiên cứu của đề tài

## Đối tượng và phạm vi nghiên cứu

## Phương pháp nghiên cứu

## Kết cấu của đề tài

CHƯƠNG 1: TIẾP NHẬN VÀ XỬ LÝ YÊU CẦU

## 1.1. Bản mô tả yêu cầu

...

## 1.2. Yêu cầu chức năng nghiệp vụ

## 1.3. Yêu cầu kỹ thuật

-Các yêu cầu kỹ thuật được xem xét và đưa ra dựa trên các khảo sát về cơ sở hạ tầng ứng dụng hiện tại của các nhà cung cấp dịch vụ liên quan và hạ tầng trang thiết bị hiện có của trường.

### 1.3.1. Yêu cầu chung về hệ thống thông tin

-Triển khai được trên hệ thống mạng nội bộ của trường hoặc trên mạng Internet toàn cầu hiện hành.

-Tương thích và vận hành tốt trên các giao thức mạng phổ biến hiện tại như: IP (IPv4, IPv6), FTP (Dùng trong tải lên/xuống các tập tin), HTTP 1.1 (Dùng trong gửi nhận dữ liệu), SMTP-IMAP (Dùng trong gửi và nhận email),...

-Có khả năng làm việc khi không có mạng, đồng bộ dữ liệu lên máy chủ tập trung khi có mạng.

-Đảm bảo tính nhất quán về mặt dữ liệu giữa các ứng dụng trên các nền tảng khác nhau, ví dụ: giữa ứng dụng Desktop và ứng dụng Web.

-Yêu cầu chung về bảo mật dữ liệu và an toàn thông tin

+ Dữ liệu phải có khả năng sao lưu và dễ dàng khôi phục lại khi cần thiết.

+ Dữ liệu chỉ được truy cập bởi những người dùng có thẩm quyền.

+ Các nghiệp vụ liên quan đến bảo mật:

* Mã hóa mã nguồn, chống dịch ngược.
* Mã hóa các thông tin nhạy cảm (mật khẩu người dùng, cấu hình cài đặt, ...) theo tiêu chuẩn mã hóa dữ liệu Advanced Encryption Standard (AES) do Viện Tiêu chuẩn và Công nghệ quốc gia Hoa Kỳ (National Institute Standards and Technology – NIST) phát hành ngày 26/11/2001.
* Các dữ liệu truyền tải giữa máy chủ và máy tính cá nhân người dùng cần đảm bảo độ an toàn bằng cách mã hóa trước khi truyền qua mạng.

### 1.3.2. Yêu cầu cụ thể

#### 1.3.2.1. Yêu cầu phần cứng

-Vận hành tốt trên các hệ thống máy tính hiện có của trường, cấu hình hệ thống máy tính tối thiểu đề nghị:

+CPU: Xung nhịp 1.0 Ghz hoặc cao hơn.

+RAM: Dung lượng 512 MB hoặc cao hơn.

+Đĩa cứng: Dung lượng trống tối thiểu 5GB

+Hiển thị tốt trên các màn hình kích cỡ 15 inch hoặc lớn hơn, độ phân giải 1024 x 768 hoặc cao hơn.

#### 1.3.2.2. Yêu cầu phần mềm

- Ứng dụng chạy trên máy tính cá nhân PC (Windows Desktop Application)

+ Hệ điều hành Windows ≥ 7.

+ Tương thích .NET Framework ≥ 4 bản đầy đủ (Full) - vì mỗi phiên bản .NET nhiều bản phân phối, ví dụ: .NET Client Profile.

+ Hệ quản trị CSDL SQL Server ≥ 2008 (chỉ yêu cầu khi sử dụng ở chế độ không có mạng).

- Ứng dụng Web (Web Desktop Application)

+Truy cập từ trình duyệt thông qua địa chỉ website.

+Chạy tốt trên trình duyệt Chrome ≥ 28, Firefox ≥ 29, Internet Explorer (IE) ≥ 8.

+Hỗ trợ thiết bị di động

+Vận hành tốt trên máy chủ IIS ASP.NET >= 4.0

CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

## 2.1. Hướng phân tích và thiết kế

-Do một số bước phân tích thiết kế có liên quan trực tiếp đến bước thực thi phần mềm nên thứ tự một vài bước phân tích cũng như cách tiếp cận sẽ bám sát và có liên hệ đến phần thực thi. Cụ thể các sơ đồ lớp được thiết kế dựa trên ngôn ngữ C#, thứ tự bước phân tích các sơ đồ lớp và các sơ đồ CSDL mức vật lý được đảo ngược trình tự để phù hợp.

-Xem chương thực thi để biết thêm các công nghệ được áp dụng trong đề tài này.

## 2.1. Lược đồ quan niệm

## 2.2. Lược đồ trường hợp sử dụng (Use cases)

-...

## 2.3. Lược đồ lớp (class)

### 2.3.1. Lược đồ các lớp thực thể

\*Ghi chú:

-Kiểu dữ liệu có dấu ? liền sau cho biết kiểu dữ liệu này chấp nhận giá trị null (trong trường hợp kiểu dữ liệu dạng nguyên thủy (primitive) hoặc kiểu cấu trúc (struct), ví dụ kiểu int, long, DateTime,...).

-Kiểu dữ liệu "T" được sử dụng trong kế thừa để lớp cha hiểu và nhận diện đúng lớp kế thừa (parameterized).

-Các thuộc tính kế thừa sẽ không liệt kê lại ở các lớp dẫn xuất để tránh dài dòng (xem chi tiết ở lớp cha).

-Các phương thức:

+Được liệt kê cuối cùng ở mỗi bảng

+Được đặc trưng bởi cặp ngoặc tròn ở sau tên phương thức.

+Chỉ liệt kê tên phương thức (không liệt kê tham số truyền vào).

+Chỉ liệt kê các phương thức quan trọng, chính yếu.

#### 2.3.1.1. Các giao diện (interface) và lớp ảo (abstract)

1. \_CRUDInterface<T>

-Mức truy cập: public

-Loại: giao diện

-Mô tả: Các lớp cần các phương thức thông dụng trên một thực thể: Thêm, Xóa, Sửa,... sẽ thực thi lớp giao diện này.

|  |  |  |  |
| --- | --- | --- | --- |
| Tên phương thức | Kiểu tham số | Kiểu trả về | Mô tả |
| add() |  | int | Thêm đối tượng vào CSDL |
| update() |  | int | Sửa đối tượng trong CSDL |
| delete() |  | int | Xóa đối tượng khỏi CSDL |
| reload() |  | T | Tải lại đối tượng |
| trigger() |  | void | Ép lazyloading tải dữ liệu ngay |
| doTrigger() |  | void | Gọi trigger trên tât cả các thuộc tính là đối tượng đơn và là khóa ngoại |
| clone() |  | T | Sao chép thông tin ra đối tượng mới |
| moveUp() |  | void | Di chuyển thứ tự đối tượng lên 1 bậc |
| moveDown() |  | void | Di chuyển thứ tự đối tượng xuống 1 bậc |
| niceName() |  | String | Lấy thông tin cơ bản về đối tượng |
| prevObj() |  | T | Trả về đối tượng có thứ tự trước 1 bậc |
| nextObj() |  | T | Trả về đối tượng có thứ tự sau 1 bậc |

2. \_EFEventRegisterInterface

-Mức truy cập: public

-Loại: giao diện

-Mô tả: Các lớp thực thể muốn đăng ký gọi ngược (callback) bởi các sự kiện khi EF làm việc sẽ thực thi lớp giao diện này.

|  |  |  |  |
| --- | --- | --- | --- |
| Tên phương thức | Kiểu tham số | Kiểu trả về | Mô tả |
| onBeforeAdded |  | void | Được gọi trước khi thêm vào CSDL |
| onAfterAdded |  | void | Được gọi sau khi thêm vào CSDL |
| onBeforeUpdated |  | void | Được gọi trước khi cập nhật vào CSDL |
| onAfterUpdated |  | void | Được gọi sau khi cập nhật vào CSDL |
| onBeforeDeleted |  | void | Được gọi trước khi xóa khỏi CSDL |
| \*Không có sự kiện "onAfterDeleted" vì khi đối tượng đã bị loại khỏi hệ giám sát của EF thì sẽ không khả dụng để gọi ngược | | | |

3. \_EntityAbstract1<T>

-Mức truy cập: public

-Loại: lớp ảo

-Lớp cha: Không có

-Lớp giao diện thực thi: \_EFEventRegisterInterface, \_CRUDInterface<T>

-Mô tả: Các lớp thực thể bắt buộc phải có kế thừa trực tiếp hoặc gián tiếp lớp này. Vì đây là các thuộc tính cơ bản cần phải có.

-Thuộc tính và phương thức

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mức  truy  cập | Loại  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public |  | id | Guid | Khóa chính |
| public |  | subId | string | Mã phụ |
| public |  | date\_create | DateTime? | Ngày tạo |
| public |  | date\_modified | DateTime? | Ngày cập nhật gần nhất |
| public |  | mota | String | Mô tả |
| public |  | order | long? | Dùng để sắp xếp thứ tự |
| protected | static | db | OurDBContext | Truy xuất đến lớp cung cấp dữ liệu |
| public | static | USNAME | String | Tên tiếng anh của lớp này |
| public | static | VNNAME | String | Tên tiếng việt của lớp này |

4. \_EntityAbstract2<T>

-Mức truy cập: public

-Loại: lớp ảo

-Lớp cha: \_EntityAbstract1<T>

-Lớp giao diện thực thi: Kế thừa từ lớp cha

-Mô tả: Các lớp có mối quan hệ n-n với hình ảnh sẽ kế thừa lớp này.

-Thuộc tính và phương thức

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mức  truy  cập | Loại  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public |  | hinhanhs | ICollection<HinhAnh> |  |

5. \_EntityAbstract3<T>

-Mức truy cập: public

-Loại: lớp ảo

-Lớp cha: \_EntityAbstract1<T>

-Lớp giao diện thực thi: Kế thừa từ lớp cha

-Mô tả: Các lớp cần xác thực đăng nhập sẽ kế thừa lớp này.

-Thuộc tính và phương thức

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mức  truy  cập | Loại  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public |  | hoten | String | Họ tên đầy đủ |
| public |  | username | String | Tên đăng nhập |
| public |  | password | String | Mật khẩu dạng băm |
| private |  | canUseUsername() | Boolean | Kiểm tra có sử dụng tên đăng nhập được chỉ định |
| public |  | canView() | Boolean | Xét quyền xem |
| public |  | canEdit() | Boolean | Xét quyền sửa |
| public |  | canAdd() | Boolean | Xét quyền thêm |
| public |  | canDelete() | Boolean | Xét quyền xóa |
| public |  | canDo() | Boolean | Xét quyền cố định |
| private |  | isRoot() | Boolean | Xét quyền root |

#### 2.3.1.2. Các lớp thực thể

1. CoSo

-Mức truy cập: public

-Loại: lớp cứng

-Lớp cha: \_EntityAbstract2<CoSo>

-Lớp giao diện thực thi: Kế thừa từ lớp cha

-Thuộc tính và phương thức

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mức  truy  cập | Loại  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public |  | ten | String | Tên cơ sở |
| public |  | diachi | String | Địa chỉ |
| public |  | days | ICollection<Dayy> |  |
| public |  | vitris | ICollection<ViTri> |  |
| public |  | permissions | ICollection<Permission> |  |

2. Dayy

-Mức truy cập: public

-Loại: lớp cứng

-Lớp cha: \_EntityAbstract2<Dayy>

-Lớp giao diện thực thi: Kế thừa từ lớp cha

-Thuộc tính và phương thức

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mức  truy  cập | Loại  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public |  | ten | String | Tên dãy |
| public |  | coso | CoSo | Cơ sở trực thuộc |
| public |  | tangs | ICollection<Tang> |  |
| public |  | vitris | ICollection<ViTri> |  |
| public |  | permissions | ICollection<Permission> |  |

3. Tang

-Mức truy cập: public

-Loại: lớp cứng

-Lớp cha: \_EntityAbstract2<Tang>

-Lớp giao diện thực thi: Kế thừa từ lớp cha

-Thuộc tính và phương thức

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mức  truy  cập | Loại  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public |  | ten | String | Tên tầng |
| public |  | day | Dayy | Dãy trực thuộc |
| public |  | vitris | ICollection<ViTri> |  |
| public |  | permissions | ICollection<Permission> |  |

4. ViTri

-Mức truy cập: public

-Loại: lớp cứng

-Lớp cha: \_EntityAbstract1<ViTri>

-Lớp giao diện thực thi: Kế thừa từ lớp cha

-Thuộc tính và phương thức

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mức  truy  cập | Loại  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public |  | coso | CoSo |  |
| public |  | day | Dayy |  |
| public |  | tang | Tang |  |
| public |  | phongs | ICollection<Phong> |  |

5. Phong

-Mức truy cập: public

-Loại: lớp cứng

-Lớp cha: \_EntityAbstract2<Phong>

-Lớp giao diện thực thi: Kế thừa từ lớp cha

-Thuộc tính và phương thức

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mức  truy  cập | Loại  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public |  | ten | String | Tên phòng |
| public |  | nhanvienpt | NhanVienPT | Nhân viên phụ trách được phân công mở cửa phòng |
| public |  | quantrivien | QuanTriVien | Quản trị viên được phân công quản lý (khác với phân quyền) |
| public |  | vitri | ViTri |  |
| public |  | permissions | ICollection<Permission> |  |
| public |  | ctthietbis | ICollection<CTThietBi> |  |
| public |  | logthietbis | ICollection<LogThietBi> |  |
| public |  | sucophongs | ICollection<SuCoPhong> |  |

6. ThietBi

-Mức truy cập: public

-Loại: lớp cứng

-Lớp cha: \_EntityAbstract2<ThietBi>

-Lớp giao diện thực thi: Kế thừa từ lớp cha

-Thuộc tính và phương thức

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mức  truy  cập | Loại  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public |  | ten | String | Tên thiết bị |
| public |  | loaithietbi | LoaiThietBi | Thuộc loại thiết bị |
| public |  | ngaymua | DateTime? | Ngày mua |
| public |  | ctthietbis | ICollection<CTThietBi> |  |
| public |  | logthietbis | ICollection<LogThietBi> |  |

7. CTThietBi

-Mức truy cập: public

-Loại: lớp cứng

-Lớp cha: \_EntityAbstract2<CTThietBi>

-Lớp giao diện thực thi: Kế thừa từ lớp cha

-Thuộc tính và phương thức

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mức  truy  cập | Loại  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public |  | soluong | int | Số lượng |
| public |  | phong | Phong | Thuộc phòng |
| public |  | thietbi | ThietBi | Thuộc thiết bị |
| public |  | tinhtrang | TinhTrang | Thuộc tình trạng |
| public |  | ngay | DateTime? | Ngày lắp vào phòng |

8. LogThietBi

-Mức truy cập: public

-Loại: lớp cứng

-Lớp cha: \_EntityAbstract2<LogThietBi>

-Lớp giao diện thực thi: Kế thừa từ lớp cha

-Thuộc tính và phương thức

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mức  truy  cập | Loại  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public |  | soluong | int | Số lượng |
| public |  | phong | Phong | Thuộc phòng |
| public |  | thietbi | ThietBi | Thuộc thiết bị |
| public |  | tinhtrang | TinhTrang | Thuộc tình trạng |
| public |  | quantrivien | QuanTriVien | Quản trị viên phát sinh |

9. SuCoPhong

-Mức truy cập: public

-Loại: lớp cứng

-Lớp cha: \_EntityAbstract2<SuCoPhong>

-Lớp giao diện thực thi: Kế thừa từ lớp cha

-Thuộc tính và phương thức

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mức  truy  cập | Loại  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public |  | ten | String | Tên sự cố |
| public |  | ngay | DateTime | Ngày phát sinh |
| public |  | phong | Phong | Thuộc phòng |
| public |  | tinhtrang | TinhTrang | Thuộc tình trạng |
| public |  | logsucophongs | ICollection<LogSuCoPhong> |  |

10. LogSuCoPhong

-Mức truy cập: public

-Loại: lớp cứng

-Lớp cha: \_EntityAbstract2<LogSuCoPhong>

-Lớp giao diện thực thi: Kế thừa từ lớp cha

-Thuộc tính và phương thức

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mức  truy  cập | Loại  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public |  | quantrivien | QuanTriVien | Quản trị viên phát sinh |
| public |  | sucophong | SuCoPhong | Thuộc phòng |
| public |  | tinhtrang | TinhTrang | Thuộc tình trạng |

11. TinhTrang

-Mức truy cập: public

-Loại: lớp cứng

-Lớp cha: \_EntityAbstract1<TinhTrang>

-Lớp giao diện thực thi: Kế thừa từ lớp cha

-Thuộc tính và phương thức

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mức  truy  cập | Loại  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public |  | type | int? | Mã loại tình trạng:  + TYPE\_THIETBI: dành cho thiết bị  + TYPE\_SUCOPHONG: dành cho sự cố phòng |
| public |  | key | String | Mã tình trạng (ví dụ: dangsudung) |
| public |  | value | String | Tên tình trạng (ví dụ: Đang sử dụng) |
| public | static | TYPE\_THIETBI | int | Giá trị hằng: 0 |
| public | static | TYPE\_SUCOPHONG | int | Giá trị hằng: 1 |
| public |  | sucophongs | ICollection  <SuCoPhong> |  |
| public |  | logsucophongs | ICollection  <LogSuCoPhong> |  |
| public |  | ctthietbis | ICollection  <CTThietBi> |  |
| public |  | logthietbis | ICollection  <LogThietBi> |  |

12. PhieuMuonPhong

-Mức truy cập: public

-Loại: lớp cứng

-Lớp cha: \_EntityAbstract1<PhieuMuonPhong>

-Lớp giao diện thực thi: Kế thừa từ lớp cha

-Mô tả: Theo mẫu hiện hành của ban Hạ tầng cơ sở.

-Thuộc tính và phương thức

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mức  truy  cập | Loại  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public |  | donvi | String | Tên đơn vị mượn (ví dụ: Khoa CNTT) |
| public |  | ngaymuon | DateTime | Thời gian mượn |
| public |  | ngaytra | DateTime | Thời gian trả |
| public |  | lydomuon | String | Lý do mượn |
| public |  | ghichu | String | Ghi chú dành cho quản trị viên khi duyệt phiếu mượn |
| public |  | trangthai | DateTime | Mã trạng thái:  + \_MOI: Chưa xử lý (mới)  + \_CHAPNHAN: Đã được duyệt (chấp nhận)  + \_TUCHOI: Đã bị từ chối (bị hủy) |
| public |  | lop | String | Lớp mượn (vd: DCT1103) |
|  |  | soluongsv | int | Số lượng sinh viên trên một phòng |
| public |  | sophong | int | Số phòng cần mượn |
| public |  | nguoimuon | QuanTriVien | Người mượn |
| public |  | nguoiduyet | QuanTriVien | Người duyệt |
| public | static | \_MOI | int | Giá trị hằng: 0 |
| public | static | \_CHAPNHAN | int | Giá trị hằng: 1 |
| public | static | \_TUCHOI | int | Giá trị hằng: -1 |

13. NhanVienPT

-Mức truy cập: public

-Loại: lớp cứng

-Lớp cha: \_EntityAbstract2<NhanVienPT>

-Lớp giao diện thực thi: Kế thừa từ lớp cha

-Thuộc tính và phương thức

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mức  truy  cập | Loại  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public |  | gioitinh | Boolean | Giới tính:  + true: Nam  + false: Nữ |
| public |  | hoten | String | Họ tên đầy đủ |
| public |  | sodienthoai | String | Số điện thoại |
| public |  | phongs | ICollection<Phong> | Danh sách phòng được phân công |

14. HinhAnh

-Mức truy cập: public

-Loại: lớp cứng

-Lớp cha: \_EntityAbstract1<HinhAnh>

-Lớp giao diện thực thi: Kế thừa từ lớp cha

-Thuộc tính và phương thức

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mức  truy  cập | Loại  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public |  | path | String | Đường dẫn tương đối tính từ thư mục chứa hình |
| public |  | cosos | ICollection<CoSo> | Danh sách cơ sở chứa hình này |
| public |  | days | ICollection<Dayy> | Danh sách dãy chứa hình này |
| public |  | tangs | ICollection<Tang> | Danh sách tầng chứa hình này |
|  |  |  |  |  |
| public |  | phongs | ICollection<Phong> | Danh sách phòng chứa hình này |
| public |  | nhanvienpts | ICollection<NhanVienPT> | Danh sách nhân viên phụ trách chứa hình này |
| public |  | sucophongs | ICollection<SuCoPhong> | Danh sách sự cố phòng chứa hình này |
| public |  | logsucophongs | ICollection  <LogSuCoPhong> | Danh sách log sự cố phòng chứa hình này |
| public |  | thietbis | ICollection<ThietBi> | Danh sách thiết bị chứa hình này |
| public |  | ctthietbis | ICollection<CTThietBi> | Danh sách chi tiết thiết bị chứa hình này |
| public |  | logthietbis | ICollection  <LogThietBi> | Danh sách log thiết bị chứa hình này |

15. LogHeThong

-Mức truy cập: public

-Loại: lớp cứng

-Lớp cha: \_EntityAbstract1<LogHeThong>

-Lớp giao diện thực thi: Kế thừa từ lớp cha

-Thuộc tính và phương thức: Sử dụng các thuộc tính ở lớp cha

16. QuanTriVien

-Mức truy cập: public

-Loại: lớp cứng

-Lớp cha: \_EntityAbstract3<QuanTriVien>

-Lớp giao diện thực thi: Kế thừa từ lớp cha

-Thuộc tính và phương thức

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mức  truy  cập | Loại  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public |  | donvi | String | Tên đơn vị |
| public |  | email | String | Địa chỉ email |
| public |  | group | Group | Thuộc nhóm quyền |
| public |  | logsucophongs | ICollection<LogSuCoPhong> |  |
| public |  | logthietbis | ICollection<LogThietBi> |  |
| public |  | phongs | ICollection<Phong> | Danh sách phòng phụ trách |

17. LoaiThietBi

-Mức truy cập: public

-Loại: lớp cứng

-Lớp cha: \_EntityAbstract1<LoaiThietBi>

-Lớp giao diện thực thi: Kế thừa từ lớp cha

-Thuộc tính và phương thức

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mức  truy  cập | Loại  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public |  | ten | String | Tên loại |
| public |  | loaichung | Boolean | Loại chung hay riêng:  + true: chung  +false: riêng |
| public |  | parent | LoaiThietBi | Loại thiết bị cha |
| public |  | childs | ICollection<LoaiThietBi> | Danh sách loại thiết bị con |

18. Group

-Mức truy cập: public

-Loại: lớp cứng

-Lớp cha: \_EntityAbstract1<Group>

-Lớp giao diện thực thi: Kế thừa từ lớp cha

-Thuộc tính và phương thức

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mức  truy  cập | Loại  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public |  | key | String | Mã nhóm quyền |
| public |  | ten | String | Tên nhóm quyền |
| public |  | permissions | ICollection<Permission> | Danh sách quyền của nhóm |
| public |  | quantriviens | ICollection<QuanTriVien> | Danh sách quản trị viên thuộc nhóm |

19. Permission

-Mức truy cập: public

-Loại: lớp cứng

-Lớp cha: \_EntityAbstract1<Permission>

-Lớp giao diện thực thi: Kế thừa từ lớp cha

-Thuộc tính và phương thức

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mức  truy  cập | Loại  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public |  | key | String | Mã quyền |
| public |  | stand\_alone | String | Quyền đơn hay quyền kết hợp |
| public |  | allow\_or\_deny | String | Quyền cho phép hay cấm |
| public |  | recursive\_to\_child | Boolean | Có áp đặt lên đối tượng liên quan |
| public |  | can\_view | Boolean | Quyền xem |
| public |  | can\_edit | Boolean | Quyền sửa |
| public |  | can\_delete | Boolean | Quyền xóa |
| public |  | can\_add | Boolean | Quyền thêm |
| public |  | cosos | ICollection<CoSo> | Danh sách cơ sở quyền này áp đặt lên (nếu có) |
| public |  | days | ICollection<Dayy> | Danh sách dãy quyền này áp đặt lên (nếu có) |
| public |  | tangs | ICollection<Tang> | Danh sách tầng quyền này áp đặt lên (nếu có) |
| public |  | phongs | ICollection<Phong> | Danh sách phòng quyền này áp đặt lên (nếu có) |
| public |  | groups | ICollection<Group> | Danh sách nhóm quyền chứa quyền này |

20. Setting

-Mức truy cập: public

-Loại: lớp cứng

-Lớp cha: \_EntityAbstract1<Setting>

-Lớp giao diện thực thi: Kế thừa từ lớp cha

-Thuộc tính và phương thức

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mức  truy  cập | Loại  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public |  | key | String | Mã cấu hình |
| public |  | value | String | Giá trị |

### 2.3.1. Lược đồ các lớp thư viện liên quan

-Ngoài các lớp thực thể chính thì xoay quanh còn có lớp các thư viện hỗ trợ, các trình trợ giúp (helper), cấu hình liên quan...

\*Vì lý do số lượng lớp thư viện quá nhiều nên sau đây chỉ liệt kê tên và mô tả

-Trình trợ giúp CSDL (Không gian tên: PTB.Global):

+client\_database: Cung cấp các phương thức giao tiếp với CSDL ở máy chủ địa phương.

+server\_database: Cung cấp các phương thức giao tiếp với CSDL ở máy chủ tập trung tập trung.

+working\_database: Cung cấp các phương thức giao tiếp với CSDL không phụ thuộc máy chủ.

-Trình trợ giúp (không gian tên: PTB.Libraries): DatabaseHelper, EmailHelper, ImageHelper, PermissionHelper.

-Trình trợ giúp (không gian tên: SHARED.Libraries): DatabaseHelper, DateTimeHelper, EmailHelper, FileHelper, FTPHelper, HTTPHelper, ImageHelper, MobileDetect, PermissionHelper, ReportHelper, ServerTimeHelper, SkinHelper, StringHelper, Debug, GUID.

-Trình cấu hình (không gian tên: PTB.Global):

+remote\_setting: Các cấu hình áp đặt cho toàn máy trạm

* ftp\_host: Các cấu hình để tải các tập tin lên máy chủ
* http\_host: Các cấu hình để tải tập tin từ máy chủ về
* smtp\_config: Các cấu hình gửi mail
* email\_template: Các cấu hình cài đặt template cho mail

+local\_setting: Các cấu hình áp đặt lên máy trạm hiện hành

+sync: Các cấu hình đồng bộ CSDL

## 2.4. Lược đồ CSDL mức vật lý

-Các lược đồ CSDL sau đây được thể hiện trên hệ quản trị SQL Server 2008 R2.

...

## 2.5. Lược đồ tuần tự (sequences)

-Do hạn chế về không gian trình bày cũng như không thể đưa hết mọi bản thiết kế vào trong báo cáo này nên nhóm chọn ra 5 chức năng chính để trình bày.

-Các sơ đồ sau được thể hiện tối đa tới mức 2 tính từ lời gọi hàm đầu tiên.

### 2.5.1.

### 2.5.2.

### 2.5.3.

### 2.5.4.

### 2.5.5.

CHƯƠNG 3: THỰC THI

## 3.1. Môi trường lập trình và phát triển ứng dụng

-Hệ điều hành: Windows 7

-Ngôn ngữ lập trình: C#

-Nền tảng .NET Framework.

-Công cụ hỗ trợ soạn thảo và biên dịch: Visual Studio 2012

-Hệ quản trị CSDL SQL Server Express 2008.

-Máy chủ web IIS 8.0 Express.

-Trình quản lý mã nguồn (Source control): Git từ nhà cung cấp Github, Inc

## 3.2. Mô hình triển khai và cách tổ chức ứng dụng

### 3.2.1. Mô hình đa nền tảng (cross-platform) trong ứng dụng hệ .NET

-Mô hình quan niệm

-Mô hình chi tiết

### 3.2.2. Mô hình 3 lớp trong lập trình hướng đối tượng (OOP)

### 3.2.3. Mô hình MVP (Model-View-Presenter) Winform dành cho ứng dụng Desktop

-Các tính năng MVP :

+Tận dụng các giao diện sử dụng chung.

+Tận dụng các logic về giao diện chung.

+Tạo sự thống nhất trong giao diện giữa các project GUI.

-Custom User control: Coi như là một module MVP nhỏ sử dụng chung cho nhiều khu vực

### 3.2.4. Mô hình ASP.NET Webform dành cho ứng dụng Web

## 3.3. Các công nghệ và kỹ thuật lập trình được áp dụng

### 3.3.1. Công nghệ Entity Framework (EF) trong lập trình dữ liệu hướng đối tượng (OOP)

#### 3.3.1.1. Tổng quan

-Là một framework giúp ánh xạ cơ sở dữ liệu quan hệ lên một khung nhìn các đối tượng và tập hợp các đối tượng.

-Là một DB ORM (Database Object-Relational Mapping) mã nguồn mở được chính Microsoft phát triển dựa trên nền tảng .NET.

-EF giúp người lập trình tiết kiệm được rất nhiều thời gian và công sức trong việc truy xuất và làm việc với các CSDL quan hệ, bởi vì bản thân EF đã cung cấp đầy đủ các tính năng và công nghệ tiên tiến giúp độc lập hóa khối CSDL ra khỏi lớp truy xuất.

-Được ứng dụng nhiều trong các nền tảng dạng Domain-Driven Design, cho phép người lập trình ảo hóa CSDL vật lý, giảm sự phục thuộc vào CSDL vật lý, từ đó lập trình viên có thể làm việc trong suốt với CSDL nền, không quan tâm hệ quản trị CSDL đích hay phiên bản khác nhau, miễn là EF hỗ trợ.

-Phiên bản mới nhất: 6.1.1

-Phiên bản sử dụng trong đề tài: 6.1.1

#### 3.3.1.2. Mô hình triển khai Code first

-Thiết kế cấu trúc CSDL bằng cách định nghĩa các lớp (class) trước (Code first): là một cách tiếp cận mới trong việc thiết kế cho các ứng dụng hướng CSDL, thay vì thiết kế cấu trúc CSDL vật lý trước thì Code first cung cấp các đặc tả để người lập trình có thể định nghĩa cấu trúc CSDL bằng các class và các logic, ràng buộc trên class; tận dụng được các tính năng kế thừa trên class, sau đó EF sẽ tự động ánh xạ bản thiết kế xuống cấu trúc CSDL.

-Code first trong bản thiết kế CSDL mới hoàn toàn (Code first to new Database): EF hỗ trợ 2 giải pháp làm việc với CSDL.

+Giải pháp 1: Đối với CSDL đã có từ trước, người lập trình nếu muốn áp dụng Code first thì cần phải tìm cách đưa bản thiết kế CSDL vật lý lên mức lớp trong mã nguồn, sau đó chỉ định bản thiết kế lớp này ứng với cấu trúc CSDL vật lý hiện tại. Kể từ thời điểm này, thiết kế sẽ được thực hiện trên mức lớp trước, sau đó sẽ được ánh xạ lại trong cấu trúc CSDL vật lý.

+Giải pháp 2: Nếu chưa có bản thiết kế CSDL vật lý săcn, thì lập trình viên chỉ cần tạo bản thiết kế lớp trong mã nguồn như mong muốn, sau đó gọi trình ánh xạ của EF để tạo cấu trúc CSDL vật lý. Đề tài này do được nghiên cứu mới hoàn toàn nên giải pháp 2 được áp dụng.

#### 3.3.1.3. Ánh xạ TPC (Table Per Concrete class) trong kế thừa thuộc tính

-Định nghĩa: TPC là cách thiết kế mà trong đó mỗi lớp thực (lớp cứng) sẽ ánh xạ thành một bảng trong CSDL vật lý. Tận dụng được các đoạn mã logic bằng cách thiết kế mô hình kế thừa, đa hình.

-Trong TPC, các lớp ảo (abstract) sẽ không được ánh xạ xuống CSDL vật lý, các lớp này chỉ dùng cho mục đích kế thừa. Muốn ánh xạ một thực thể xuống CSDL vật lý thì lớp thực thể đó bắt buộc không được khai báo là một lớp ảo.

#### 3.3.1.4. Tích Đề-các (Descartes) trong biểu diễn quan hệ 1-n hoặc n-n

-Gọi tập hợp A = {Table A0, Table A1,...Table Am} chứa các thực thể ở quan hệ nhiều.

-Gọi tập hợp B = {Table B0, Table B1, ... Table Bn} chứa các thực thể ở quan hệ 1.

-Một tích Đề-các diễn tả mối quan hệ 1-n hoặc n-n từ Bi đến Aj (i < [B], j < [A])

B x A { (b,a) | b thuộc B, a thuộc A }.

-Mối quan hệ 1-n trong CSDL quan hệ có thể được biểu diễn bằng 2 cách sau:

+Cách 1: Nâng cấp quan hệ 1-n thành quan hệ n-n và ngầm định không sử dụng chiều tham chiếu ngược lại: Khi đó mỗi quan hệ (B x A)i ( i < [B x A] ) sẽ liên kết với (1+1)+1=3 Table vật lý.

=> Tổng các bảng vật lý tối thiểu cần thiết để biểu diễn quan hệ trên (AxB) là:

[A] + [B] + [B x A].

Phù hợp với trường hợp mối quan hệ cần có thêm các thuộc tính kèm theo.

+Cách 2: Truyền thống, đối tượng ở quan hệ nhiều sẽ có n khóa ngoại trỏ đến đối tượng ở quan hệ 1: Khi đó mỗi quan hệ (B x A)i (i < [B x A]) sẽ liên kết ứng với 1+1=2 bảng vật lý

=> Tổng các bảng vật lý tối thiểu cần thiết để biểu diễn quan hệ trên (B x A) là:

[A] + [B]

\*EF làm việc được với cả 2 cách biểu diễn trên. Tùy thuộc vào từng trường hợp cụ thể mà lựa chọn phương án thích hợp.

-Mối quan hệ n-n trong CSDL quan hệ có thể được biểu diễn bằng 2 cách:

+Cách 1: tương tự cách 1 trong biểu diễn quan hệ 1-n (nhưng vì đã là quan hệ n-n nên không cần nâng cấp).

+Cách 2: Sử dụng thuộc tính đa trị trong từng trường định nghĩa khóa ngoại, tuy nhiên sẽ vi phạm dạng chuẩn CSDL 1NF (dạng chuẩn thấp nhất) do chứa thuộc tính đa trị, và không được EF hỗ trợ, nên không khả thi khi triển khai.

\*Đồ án sử dụng cách 1 trong việc biểu diễn quan hệ n-n giữa: Cơ sở, Dãy, Tầng, Phòng,... và Hình Ảnh

#### 3.3.1.5. Công nghệ truy vấn LINQ

- LINQ to Entities: EF sử dụng LINQ để truy vấn trên CSDL. LINQ là một công nghệ truy vấn CSDL không phụ thuộc vào dữ liệu nguồn, tức là có thể dùng LINQ để truy vấn và thực thi các thao tác dữ liệu trên: tập hợp/danh sách các đối tượng, tập tin XML hay các hệ quản trị CSDL khác nhau.

-LINQ IEnumberable và biểu thức chính quy Lambda

+LINQ IEnumberable: là một lớp giao diện (interface) định nghĩa trong LINQ, cho phép tạo ra các câu truy vấn lồng nhau trên một tập hợp có thực thi lớp giao diện này. Kết quả của một loạt các thực thi sẽ được trả về ngay sau khi có lời gọi để chuyển từ tập hợp không chính quy (non-generic) sang tập hợp chính quy (generic). Đây là tính năng rất hay trong LINQ, mà nhờ đó tiết kiệm được chi phí cũng như thời gian thực thi dữ liệu, vì câu truy vấn cuối cùng chỉ được kết lại và gọi chạy khi hoàn tất một loạt các truy vấn lồng nhau thay vì phải tải dữ liệu lên sau mỗi bước nạp điều kiện.

+Lambda là một dạng biểu thức chính quy được sử dụng trong các truy vấn đến các lớp dữ liệu có đăng ký thực thi lớp IEnumberable.



Hình 3.x: Cách hoạt động của kỹ thuật truy vấn lồng

#### 3.3.1.6. Các tính năng khác được áp dụng trong đề tài

-Trì hoãn tải dữ liệu (Lazy loading): Trì hoãn tải dữ liệu khi chưa cần thiết, dữ liệu chỉ được tải khi có lời gọi tới lần đầu tiên (Giảm thời gian nạp dữ liệu ban đầu). Lazy loading rất hữu ích trong trường hợp một các đối tượng có mối quan hệ phức tạp qua lại với nhau và cây liên hệ có chiều cao lớn hơn 1. Mặc định EF được cấu hình bật Lazy loading.



Hình 3.x: Tham chiếu ngược trên các quan hệ 1-n, n-n

-Phiên bản CSDL (Database Version): (Upgrade/Downgrade/Rebase): EF xem mỗi sự thay đổi trong bản thiết kế lớp (class) sẽ tương ứng với một phiên bản CSDL mức vật lý, khi bản thiết kế có sự thay đổi, EF sẽ ghi nhận lại sự thay đổi đó và tạo ra các đoạn mã để điều chỉnh cấu trúc CSDL hiện tại lại cho khớp với bản thiết kế mới. Những đoạn mã này được gọi là một phiên bản của CSDL, và sẽ được định danh bằng tên mã nhằm phục vụ cho quá trình dịch chuyển phiên bản.

-Dịch chuyển phiên bản CSDL (Migration to database Version):

+Nâng cấp (Upgrade): là hình thức dịch chuyển từ phiên bản thấp Vi lên phiên bản cao hơn Vj (i<j).

+Hạ cấp (Downgrade): là hình thức di chuyển từ phiên bản cao Vj xuống phiên bản thấp hơn Vi (i<j).

\*EF cho phép nhảy cốc giữa các phiên bản, ví dụ có thể nhảy từ phiên bản v3 xuống v1 mà không cần thông qua v2 và ngược lại, miễn là các v1, v2, v3 đã được định danh trước đó. Tính năng này rất hữu ích khi dự án được thực hiện ở quy mô nhóm, khi đó các lập trình viên sẽ tiết kiệm được rất nhiều thời gian cho khâu cập nhật CSDL trong suốt quá trình lập trình và kiểm thử.

-Sự tương thích giữa các phiên bản CSDL: đây cũng là vẫn đề được quan tâm tới trong khi thiết kế và vận hành ứng dụng. EF chỉ có thể làm việc trên phiên bản CSDL Vi nếu Vi tương thích với phiên bản CSDL hiện tại (ứng với thiết kế lớp (class) hiện tại). Tùy thuộc vào sự thay đổi giữa các phiên bản mà có thể tương thích hoặc không tương thích cụ thể như sau:

-Xét 2 phiên bản CSDL Vi và Vj (i<j):

+Tương thích ngược: Vj được xem là tương thích ngược với Vi nếu CSDL Vj bao trùm CSDL Vi.



+Tương thích xuôi: Vi được xem là tương thích xuôi với Vj nếu CSDL Vi bao trùm CSDL Vj (điều này rất hiếm khi xảy ra vì đa phần bản thiết kế mới Vj luôn mở rộng hơn so với bản cũ Vi).



+Không tương thích: Trong các trường hợp còn lại.





\*Các ví dụ minh họa trên chỉ thể hiện ở mức đơn giản nhất, trên thực tế việc xét tính tương thích sẽ phức tạp hơn nhiều do sự kết hợp của nhiều bảng và định nghĩa các thuộc tính trên bảng.

-Kiểm định mô hình (Model checking): Kiểm tra cấu trúc CSDL có bị thay đổi bởi các tác nhân ngoài hệ thống hay không ?, đảm bảo hệ thống làm việc ổn định và đúng đắn. Model cheking là cực kỳ quan trọng khi làm việc với EF bởi vì các trường thuộc tính của dữ liệu vật lý gắn chặt với các lớp tương ứng của ứng dụng khi bộ máy ánh xạ hoạt động, nên chỉ với một thay đổi nhỏ về định nghĩa của CSDL vật lý cũng sẽ khiến EF không hoạt động.

-Trình khởi tạo CSDL tùy biến (Custom Database Initalizer): Chỉ định phương thức khởi tạo CSDL, EF cho phép lựa chọn và tùy biến nhiều chỗ trong quá trình tạo tự động cấu trúc CSDL khi ánh xạ bản thiết kế xuống CSDL mức vật lý, các tính năng bao gồm: tự động tạo CSDL nếu chưa có ?, tự động tạo bảng nếu chưa có ?, tự động tạo dữ liệu mẫu ban đầu ?,…)

+CreateDatabaseIfNotExists: mặc định của EF. Tự động tạo CSDL nếu chưa có.

+DropCreateDatabaseIfModelChanges: Tự động tạo lại CSDL khi cấu trúc bị thay đổi. Tuy nhiên không an toàn vì chỉ một sơ suất nhỏ trong khâu thiết kế cũng sẽ dẫn đến nguy cơ mất CSDL.

+DropCreateDatabaseAlways: Luôn luôn xóa và tạo mới lại CSDL mỗi khi ứng dụng khởi chạy, phù hợp với các ứng dụng sử dụng CSDL như là một bộ nhớ tạm trong lúc làm việc, và muốn CSDL rỗng cho mỗi phiên mới.

+Custom DB Initializer: Nếu các định nghĩa trên không đáp ứng được yêu cầu thì người lập trình có thể tự định nghĩa một trình khởi tạo dữ liệu riêng cho CSDL.

-Tạo dữ liệu mẫu mặc định (Data Seeding): Cung cấp các đặc tả để tạo dữ liệu mặc định ban đầu khi tạo mới một CSDL, ví dụ: Tài khoản quản trị mặc định, các giá trị cài đặt mặc định,...

-Mô hình dữ liệu hướng sự kiện (Event-Driven model - BootStrapper): tương tự như mô hình lập trình giao diện hướng sự kiện được áp dụng trong Winform, mô hình dữ liệu hướng sự kiện cũng cho phép các đối tượng tham gia đăng ký các sự kiện xảy ra khi hệ thống theo dõi của EF làm việc. Được ứng dụng trong các nghiệp vụ như: sau khi cập nhật thành công thì trường "date\_modified" sẽ có giá trị thời gian hiện tại, sau khi sửa đổi một đối tượng sẽ ghi nhật ký hệ thống...Các nghiệp vụ này sẽ được thực thi một cách tuần tự và chính xác như mong muốn của lập trình viên.



Hình 3.x: Giao tiếp 2 chiều trong mô hình dữ liệu hướng sự kiện

-Khai báo proxy cho các thuộc tính (property virtual proxy): EF đòi hỏi thuộc tính khóa ngoại của các lớp phải được khai báo dạng ảo (virtual) để EF được phép tạo proxy ẩn trong các kỹ thuật theo dõi hay Lazy loading khi các thuộc tính này được truy xuất hoặc sửa đổi.

-Trình quản lý giao dịch (Transaction Manager) trong các kỹ thuật quay ngược (rollback): Trong một ứng dụng lớn thì số tác vụ con thực thi trong một nghiệp vụ cụ thể là nhiều, hệ quản trị CSDL chỉ xử lý đơn nguyên dữ liệu (hoặc là tất cả thao tác trên dữ liệu đều được thực thi hoặc là không) ở mức rất thấp (mức tác vụ INSERT, DELETE,... trên từng đối tượng). Do đó, nếu muốn đảm bảo tính đơn nguyên ở mức cao hơn (cả một nghiệp vụ hoàn chỉnh) thì việc điều khiển và gọi trình quản lý giao dịch đúng thời điểm sẽ giải quyết được bài toán "đảm bảo toàn vẹn CSDL". EF cung cấp một cơ chế Transaction rất đơn giản và hiệu quả.

-Trạng thái của đối tượng (Entity State) và tính năng cập nhật chọn lọc: Để có thể theo dõi được các sự thay đổi dữ liệu trên các đối tượng trong lúc thực thi (runtime), EF đưa ra định nghĩa về trạng thái của các đối tượng, trong đó một đối tượng có thể thuộc một trong các trạng thái sau:

+Attached: Đối tượng mới khởi tạo và được đưa và hệ thống theo dõi, tuy nhiên chưa được lưu xuống CSDL.

+Detached: Đối tượng đã bị loại khi hệ thống theo dõi.

+Added: Đối tượng được đưa vào hàng đợi, chờ thêm vào CSDL.

+Modified: Đối tượng được đánh dấu là đã bị thay đổi ít nhất 1 thuộc tính, được đưa vào hàng đợi chờ cập nhật xuống CSDL.

+Unchanged: Đối tượng được đánh dấu là sạch, có thể là mới được khởi tạo hoặc là mới được tải lên từ CSDL.

+Deleted: Đối tượng được đánh dấu là bị xóa, được đưa vào hàng đợi chờ xóa khỏi CSDL.

-Trình lọc dữ liệu (DataFilter) trong hiển thị dữ liệu: Một đối tượng chỉ mang các thuộc tính trực tiếp của bản thân nó, trong lập trình giao diện, thông thường khi hiển thị thông tin một đối tượng nào đó, ta thường hiển thị các thuộc tính gián tiếp (thuộc tính của khóa ngoại), do đó trình xử lý giao diện sẽ không làm việc trực tiếp với các đối tượng này mà làm việc thông qua một lớp mặt nạ (mask) gọi là lớp lọc dữ liệu (DataFilter), nhiệm vụ của DataFilter là kết các đối tượng có liên quan lại với nhau sau đó chọn ra các thuộc tính cần hiển thị.

-Ngữ cảnh CSDL (Database Context) và cơ chế hoạt động lớp truy xuất CSDL ( Singleton Database Instance Provider):

+EF xem Database Context là một ngữ cảnh truy xuất đến CSDL, trên đó chứa các định nghĩa về nguồn dữ liệu. Database Context là không gian làm việc của EF. Có thể có nhiều Database Context được định nghĩa trên cùng một CSDL.

+Singleton giúp các lớp thực thể nhìn thấy cùng một DbContext trong suốt phiên làm việc, bởi vì EF đòi hỏi các đối tượng sinh ra từ các lớp thực thể phải thống nhất về DbContext, một đối tượng không thể được theo dõi bởi các DbContext khác nhau.

+Phiên làm việc được đánh dấu từ lúc DbContext được khởi tạo cho đến khi bị hủy bỏ (Dispose), các đối tượng nằm ngoài phiên làm việc được xem là không hợp lệ và không có ý nghĩa về mặt dữ liệu, muốn làm việc lại trên các đối tượng này nhất thiết phải được tải lại trong một phiên làm việc DbContext khác.

### 3.3.2. Công nghệ Sync Framework của Microsoft trong đồng bộ CSDL tập trung

#### 3.3.2.1. Tổng quan

-Định nghĩa: Sync Framework là công nghệ được Microsoft phát triển với mục đích chính là đồng bộ dữ liệu qua lại giữa các nguồn dữ liệu, dữ liệu nguồn ở đây có thể là hệ thống tập tin hoặc là một CSDL của một hệ quản trị nào đó, hiện Sync Framework hỗ trợ các dữ liệu nguồn tương thích chuẩn ADO.NET mà trong đó hệ quản trị MSSQL Server hoàn toàn đáp ứng được các yêu cầu trên.

-Đồng bộ dữ liệu giữa các CSDL với mức đơn vị dữ liệu là bảng.

-Sử dụng kỹ thuật trigger trên từng bảng, trigger có nhiệm vụ thu thập và bắt các sự thay đổi về dữ liệu trên CSDL, sau đó lưu trữ lại trong các bảng theo dõi (tracking table, mỗi bảng được chỉ định trong Sync Scope sẽ phát sinh ra một bảng theo dõi tương ứng) mà Sync Framework tạo ra khi một Sync Scope được cài đặt.



-Nhờ các kỹ thuật lưu trữ và theo dõi dữ liệu như đã được giới thiệu ở trên mà Sync Framework sẽ chỉ đồng bộ những dữ liệu sai khác giữa các dữ liệu nguồn, do đó quá trình xử lý và truyền nhận dữ liệu sẽ tiết kiệm được thời gian và tài nguyên hệ thống, khác với các giải pháp truyền thống là phải tải mới toàn bộ dữ liệu.

-Phiên bản mới nhất hiện tại: 2.1

-Phiên bản sử dụng trong đề tài: 2.1

\*Trong phạm vi ứng dụng của đề tài này, sẽ chỉ xem xét đến nguồn dữ liệu là hệ quản trị CSDL, cụ thể là MSSQL Server.

#### 3.3.2.2. Đồ thị tiến trình đồng bộ trong kỹ thuật tránh deadlock

-Đồ thị tiến trình đồng bộ: như đã đề cập ở phần trên, Sync Framework đồng bộ ở mức đơn vị dữ liệu là bảng, nên thứ tự các bảng trong một tiến trình đồng bộ là vô cùng quan trọng vì đặc thù ràng buộc khóa ngoại của CSDL quan. Nếu bảng B có chứa khóa ngoại tham chiếu đến bảng A thì bảng A phải được xử lý trước bảng B.

=> Cần phải xây dựng đồ thị tiến trình cho phiên đồng bộ.

-Đặc điểm của đồ thị tiến trình đồng bộ:

+Do thứ tự bảng trước sau được xem xét nên đồ thị là đồ thị có hướng.

+Do trong CSDL quan hệ không được phép chứa có liên hệ vòng nên đồ thị là đơn đồ thị, không chứa chu trình (vì chu trình sẽ gây chết tiến trình đồng bộ (deadlock)) và không bao giờ là một đồ thị liên thông mạnh.

+Đồ thị có thể liên thông hoặc không liên thông tùy thuộc vào sự giao nhau giữa các tập quan hệ bảng.

+Đồ thị có thể có nhiều đồ thị con (các thành phần liên thông).

+Các thành phần liên thông có thể là các cây, trường hợp này cây có thể được tách ra thành nhiều cây con để xử lý song song bằng các tiến trình song song hoặc cũng có thể được xử lý tuần tự bằng một tiến trình duy nhất, tùy thuộc thiết kế đơn luồng hay đa luồng lúc thực thi.

-Xem các bảng là các đỉnh (V - vertexes).

-Xem các liên hệ khóa ngoại từ bảng này đến bảng kia là các cung (E - edges), trong đó một cung có nút con là bảng chứa khóa ngoại và nút cha là bảng tham chiếu đến.

-Ta được đồ thị có hướng G = {V, E}, trong đó:

+Tập hợp các nút: V = {Table 0, Table 1, ...., Table n}.

+Tập hợp các cung: E = {e 0 = (V i, V j), e 1 = (V p, V q),...e m = (V t, V v)} (i,j,p,q,t,v < n).

-Nguyên tắc xây dựng tiến trình đồng bộ song song:

+Mỗi đồ thị con Hi trong đồ thị G sẽ được xử lý bởi một tiến trình Pi. Nếu Hi là cây thì Hi sẽ được tách ra thành n cây khác nhau tại vị trí nút nhánh. và kéo theo sự phát sinh n tiến trình con W tương ứng (trong đó n là bậc phát của nút nhánh). Các cây con này sẽ tiếp tực được chia tách nếu vẫn còn nhánh.

+Thời gian sống của tiến trình con W phụ thuộc vào vòng đời sống của tiến trình cha P.

+Trong mỗi tiến trình Pi, ta dùng giải thuật sau để thiết lập hàng đợi đồng bộ Qi:

* Bước 1: Khởi tạo hàng đợi Q có kích thước n phần tử, n là số đỉnh của đồ thị Hi tương ứng.
* Bước 2: Thiết lập vị trí bắt đầu BD=0 và kết thúc KT=n-1
* Bước 3: Lập danh sách các đỉnh phát, đỉnh thu của đồ thị Hi.
* Bước 4: Đưa danh sách đỉnh phát vào đầu hàng đợi kể từ vị trí BD, không quan trọng thứ tự trước sau. Đưa danh sách đỉnh thu vào cuối hàng đợi từ vị trí KT trở về trước, không quan trọng thứ tự trước sau.
* Bước 5: Xóa các đỉnh phát, đỉnh thu và các cung liên quan ra khỏi đồ thị Hi.
* Bước 6: Thiết lập lại hàng đợi Q với:

BD = BD + [tập đỉnh phát]

KT = KT - [tập đỉnh thu]

* Bước 7: Nếu hàng đợi Q rỗng (BD>=KT) thì kết thúc ngược lại thì quay lại bước 3.

+Nếu cây Ti có nhánh thì tiến hành cắt cây tại nút nhánh đó, lúc này tiến trình Pi tương ứng sẽ được chia ra thành w tiến trình Q song song (trong đó: w là bậc ngoài (số cung đi ra từ nút nhánh)), lặp lại cho các nhánh tiếp theo cho đến khi duyệt hết cây.

-Việc xây dựng tiến trình đồng bộ song song chỉ thực sự phát huy hiệu quả đối với những đồ thị có số lượng lớn các nút (cỡ khoảng trên 100 nút) và các cây (nếu có) có nhiều nhánh. Trong phạm vi đề tài này đồ thị có số lượng nút rơi vào khoảng 30-40 nút và các cây (nếu có) có ít nhánh nên việc xây dựng cây tiến trình song song sẽ không mang lại hiệu quả nhiều và rất phức tạp. Do đó phương pháp sử dụng một tiến trình duy nhất được xem xét tới.

-Nguyên tắc xây dựng tiến trình đồng bộ đơn nhất:

+Chỉ có một tiến trình P duy nhất cho toàn đồ thị.

+Các nút trong từng đồ thị sẽ lần lượt được đưa vào danh sách hàng đợi Q sao cho mệnh đề sau luôn đúng:

"Với mọi Qi, Qj thuộc Q nếu cung e=(Qi, Qj) thuộc E thì j phải nhỏ hơn i"

(tức là nếu bảng A có chứa khóa ngoại đến bảng B thì bảng B phải được xử lý đồng bộ trước bảng A)

=> Phương pháp này có thể được thực hiện bằng giải thuật sắp xếp trong đó điều kiện so sánh là xét cung tạo thành có thuộc đồ thị hay không.

-Không gian đồng bộ (Sync Scope):

+Định nghĩa: Sync Scope là được hiểu như là định nghĩa về một phiên đồng bộ trên một CSDL cụ thể, chứa các thông tin về tập hợp bảng cần đồng bộ. Một CSDL có thể có nhiều hơn một Sync Scope. Một CSDL có thể có nhiều Sync Scope được thiết lập sẵn thông qua việc mở rộng vùng lưu trữ trên CSDL bằng các bảng tạm và các procedure chức năng

+Cài đặt một Sync Scope lên CSDL có sẵn: là một loạt các thao tác cần thiết để khởi tạo và định danh một Sync Scope lên trên CSDL đã có sẵn dữ liệu hoặc CSDL mới hoàn toàn, trong đó việc chỉ định danh sách các bảng cần đồng bộ được xem là quan trọng nhất, khi các phương thức khởi tạo Sync Scope được gọi, Sync Framework sẽ thực hiện nhiệm vụ còn lại.

+Gõ bỏ một Sync Scope có sẵn ra khỏi CSDL: thao tác gỡ bỏ Sync Scope sẽ ngược lại với các bước khi cài đặt trước đó. Chỉ cần chỉ định dữ liệu nguồn và tên Sync Scope, sau đó gọi phương thức gỡ bỏ, Sync Framework sẽ thực hiện nhiệm vụ còn lại. Việc gõ bỏ Sync Scope sẽ đồng nghĩa với việc CSDL sẽ không thể cung cấp phiên đồng bộ cho các trình quản lý đồng bộ, và do đó sẽ không tham gia vào hệ thống.

+Áp đặt Sync Scope giữa các CSDL (Fetching scope among databases): là sao chép các thông tin về các định nghĩa phiên, các cấu hình liên quan,... từ một Sync Scope trên CSDL A có sẵn sang một Sync Scope mới trên CSDL mới B, khai báo rằng CSDL A và B có thể "bắt tay" được với nhau, lúc này trình quản lý đồng bộ mới có thể nhìn thấy và làm việc được trên cả 2 CSDL này, cũng có thể nói đây là bước thiết lập một cầu nối (pipeline) dữ liệu. Việc áp đặt Sync Scope từ một CSDL này lên một CSDL là rất quan trọng. Trong một tiến trình đồng bộ cụ thể thì thao tác này được thiết lập sớm nhất ngay khi có thể. Một CSDL có thể bắt tay với nhiều CSDL khác, đây là tính năng sẽ được ứng dụng trong các mô hình triển khai máy trạm khi đưa vào vận hành.

+Hướng đồng bộ lên/xuống/2 chiều, tính thông nhau giữa các cầu nối (Sync Direction (Up/Down/Bidirectional link)): khi một cầu nối được thiết lập giữa 2 CSDL thì việc quy định hướng của luồng đồng bộ được xem xét và quyết định tùy thuộc vào yêu cầu về chức năng và quyền hạn của các máy trạm. Ví dụ: trong hệ thống có những máy trạm chỉ muốn sao chép các thay đổi từ máy chủ tập trung xuống để xem và thống kê mà không có các thao tác thay đổi (read only) thì giải pháp Download only được áp dụng. Việc chọn giải pháp phù hợp sẽ làm tăng tính an toàn dữ liệu vì việc cập nhật sửa đổi đã được kiểm soát ở mức CSDL thấp hơn so với mức ứng dụng (Application), tính bảo mật dữ liệu cũng được tăng lên.

#### 3.3.2.3. Tiến trình song song trong đồ thị dạng cây

...

#### 3.3.2.4. Đụng độ dữ liệu và giải pháp khóa chính GUID

-Đụng độ dữ liệu: trong hệ đồng bộ thì các đụng độ về mặt dữ liệu là không thể tránh khỏi, đặc trưng cho tính không nhất quán trên cấu trúc CSDL (ví dụ: có nhiều hon 2 khóa chính trong cùng một bảng).

-Đụng độ vật lý trên khóa chính (primary key conflict): Xét 2 CSDL độc lập có cùng cấu trúc bảng và dữ liệu, nếu khóa chính được thiết lập dạng tự động tăng (Auto Increasement) thì khi gọi phương thức chèn mới (INSERT) trên 2 CSDL, hệ quản trị CSDL địa phương ở cả 2 CSDL trên sẽ có khả năng tạo ra khóa chính trùng nhau rất cao (do tính tuần tự trong cấp phát). Nên khi đồng bộ dữ liệu sẽ bị trùng khóa chính, 1 trong 2 dữ liệu mới đó sẽ không được hệ quản trị tiếp nhận.



=> Sử dụng khóa chính kiểu GUID (Globally Unique Identifier): GUID là kiểu giá trị "tự nhiên" (Natural), được tạo ra dựa trên sự kết hợp giữa các giá trị định danh (địa chỉ MAC của card mạng) và ngẫu nhiên (thời gian hiện tại trên hệ quản trị CSDL), được nhiều hệ thống hỗ trợ. Hệ quản trị CSDL sẽ đảm bảo mỗi GUID được cấp phát sẽ là duy nhất trên toàn cầu (mặc dù khả năng trùng là có thể xảy ra trên lý thuyết nhưng thực tế có thể chấp nhận được tùy vào phạm vi sử dụng nội bộ hay liên mạng hay toàn cầu). GUID được chia làm 2 loại lớn:

+GUID ngẫu nhiên: các GUID được tạo ra không theo một trật tự nào cả. Sử dụng trong các trường hợp bảo mật cao (do rất khó để đoán được giá trị cấp phát tiếp theo). Tuy nhiên CSDL lưu trữ GUID dạng này sẽ bị phân mảnh.

+GUID tuần tự: các GUID được tạo ra theo một trật tự. Giảm sự phân mảnh.



-Đụng độ logic do sự trễ (Delay) dữ liệu: một sự trễ dữ liệu được định nghĩa khi mà sự thay đổi về mặt CSDL ở một máy trạm khác (dù đã đẩy hay chưa đẩy lên máy chủ tập trung) chưa kịp cập nhật cho máy trạm địa phương (local machine) mà máy trạm địa phương cũng đã tạo nên một sự sửa đổi. Khi mà đụng độ về khóa chính (đụng độ vật lý) đã được giải quyết thì đụng độ logic do sự chậm trễ trong việc cập nhật dữ liệu lên máy chủ tập trung là vấn đề không thể tránh khỏi do đặc thù làm việc Offline, và Sync Framework chỉ có thể can thiệp và xử lý tự động khi hạng mục dữ liệu bị thay đổi ở cả 2 nguồn là không giao nhau hoặc hạng mục dữ liệu giao nhau là đơn nhất. Đối với các trường hợp sự thay đổi ở một hạng mục kéo theo sự thay đổi ở các hạng mục liên quan thì Sync Framework không thể can thiệp được.

-Đây cũng là cách làm việc chung cho các hệ đồng bộ hiện nay. Việc tránh đụng độ kiểu này phải do lập trình viên tự quy định các chính sách về đồng bộ dữ liệu giữa các máy trạm và máy chủ tập trung.

#### 3.3.2.5. Kỹ thuật khóa Semaphore trong xử lý đa luồng (multi thread) trên hàng đợi (queue)

-Tác vụ bất đồng bộ (asynchronous action) là các tác vụ được gọi chạy nền bằng các luồng riêng biệt mà không cần biết kết quả trả về (cách hoạt động gần giống giao thức UDP trong truyền tin không xác báo).

-Khóa semaphore giúp ngăn chặn (trong semaphore đơn tiến trình) hoặc giảm thiểu (trong semaphore đa tiến trình) sự đụng độ trong truy cập tài nguyên từ các tác vụ bất đồng bộ tạo nên bởi các luồng xử lý song song bằng cách tạo ra các chính sách về phiên và các giao tác trên hàng đợi.



Hình 3.x: Minh họa semaphore đơn tiến trình trong xử lý đa luồng trên hàng đợi

-Có thể không cần quản lý các luồng truy xuất dữ liệu song song, vì bản thân hệ quản trị CSDL đã có các chính sách về khóa trên dữ liệu, tuy nhiên lúc này sẽ gây tải lên cho cả hệ quản trị CSDL địa phương lẫn máy chủ tập trung. Chúng ta nên có giải pháp quản lý ở mức ứng dụng nhằm tránh gây ra các tải không cần thiết lên máy chủ, nhất là trong môi trường liên mạng (tiết kiệm được tài nguyên hệ thống, ví dụ: băng thông đường truyền).

-Cụ thể trong chính sách đồng bộ dữ liệu với máy chủ tập trung: Nhằm đảm bảo dữ liệu có tính sẵn sàng cao và nhất quán, các chính sách sau cần được áp dụng trong kỹ thuật xử lý đồng bộ:

+Luôn đồng bộ dữ liệu mới nhất từ máy chủ tập trung về trước khi thực hiện các sửa đổi trên CSDL (nhằm hạn chế sự trễ dữ liệu).

+Sau khi một máy trạm hoàn tất một thao tác sửa đổi trên CSDL thì phải đồng bộ lên máy chủ tập trung ngay lập tức (nhằm đảm bảo hệ thống luôn phản ánh đúng CSDL hiện tại).

### 3.3.4. Công nghệ DevExpress trong lập trình giao diện

### 3.3.5. Công nghệ giao diện tùy biến (responsive design) dành cho ứng dụng Web Mobile

Bảng 3.1. Tên bảng

Bảng 3.2. Tên bảng

CHƯƠNG 4: KIỂM THỬ

## 4.1. Kiểm thử tự động mức mã nguồn (Unit test)

### 4.1.1. Kiểm thử hộp đen (Black box testing)

-Trong Unit test có 2 phương pháp kiểm thử cơ bản là kiểm thử hộp trắng (kiểm thử luôn cả dữ liệu vào/ra và cả cách thực hiện cụ thể) và kiểm thử hộp đen (chỉ kiểm thử các bộ dữ liệu vào/ra mà không quan tâm đến cách thực hiện cụ thể).

-Do quy mô của ứng dụng không quá phức tạp nên trong phạm vi đề tài này chỉ chọn giải pháp kiểm thử hộp đen.

### 4.1.2. Mô hình kiểm thử AAA (Arrange-Act-Assert)

-Đây là trường phái kiểm thử Unit test phổ biến nhất. Trong đó:

+Sắp xếp (Arrange): là lựa chọn các thành phần tham gia kiểm thử. có thể là các biến, các lớp hoặc thậm chí là các dự án (project) ngoài.

+Act (Action - hiện thực): là mô phỏng lại các bước, thao tác trong lịch trình kiểm thử nhằm tạo ra các kết quả đầu ra (output) tương ứng với các kết quả đầu vào (input).

+Assert (đánh giá): là một so sánh giữa kết quả đầu ra thực tế và kết quả đầu ra mong muốn.

## 4.2. Kiểm thử chấp nhận (Acceptance test)

-Kiểm thử chấp nhận được thực hiện bởi người dùng hoặc lập trình viên đứng ở góc độ là người dùng.

-Giai đoạn kiểm thử này có thể được thực hiện thủ công hoặc tự động tùy thuộc vào người đứng đầu dự án kiểm thử. Do đặc tính của kiểm thử tự động đòi hỏi phải bỏ ra nhiều thời gian cho phần thiết kế hệ thống hơn là thời gian thực thi kiểm thử, nên nhóm đã quyết định chọn giải pháp kiểm thử thủ công cho giai đoạn này.

CHƯƠNG 5: VẬN HÀNH VÀ BẢO TRÌ

## 5.1. Hướng dẫn sử dụng và tài liệu kỹ thuật

-Chụp ảnh các màn hình chức năng và hướng dẫn sử dụng cho các chức năng chính: Cấu hình, quản lý

## 5.2. Vận hành hệ thống

-Cần nói được cách triển khai hệ thống để làm việc ?

-Cách thiết lập mạng lưới triển khai phần mềm

-Mô hình thiết lập theo kiểu Work Station - Data Center hay Client-Server mức đơn giản

-Các lưu ý khi triển khai hệ thống. Các lỗi thường gặp và giải pháp

## 5.3. Bảo trì

-Cần nói lên được các thủ tục và biện pháp nghiệp vụ cần thiết diễn ra trong quá trình hệ thống làm việc để đảm bảo hệ thống làm việc ổn định và an toàn.

-Hệ thống dự phòng backup-system khi có sự cố diễn ra trên main-system

-

KẾT LUẬN VÀ KIẾN NGHỊ

Nội dung kết luận....

TÀI LIỆU THAM KHẢO

Danh mục tài liệu tham khảo....